WHAT IS CLAIMED IS

1. A method comprises:

assigning tasks for packet processing to a plurality of programming engines;

establishing programming stages corresponding to the plurality of programming engines; and,

establishing a plurality of pipelines between the programming stages.

2. The method of claim 1 further comprising establishing contexts for the assigned tasks on the plurality of programming engines.

3. The method of claim 2 wherein establishing contexts for the assigned tasks comprises utilizing a software controlled cache.

4. The method of claim 3 wherein the software controlled cache is a content addressable memory (CAM).

5. The method of claim 1 further comprising forming at least one next neighbor register residing in each of the

plurality of programming engines.

6. The method of claim 5 wherein establishing the plurality of pipelines comprises transferring data from the at least one next neighbor register residing in one of the plurality of programming engines to a subsequent next neighbor register residing in an adjacent programming engine from the one programming engine.

7. The method of claim 6 wherein the one programming engine maintains the currently operating programming stage of the pipeline and the adjacent programming engine maintains a subsequent programming stage of the plurality of pipelines.

8. The method of claim 1 further comprising modifying variables in the assigned tasks utilized in the programming stages by the plurality of programming engines.

9. The method of claim 8 wherein the variables are shared variables utilized by the programming stages of the plurality of programming engines and include a critical section defining the read-modify-write time of the shared variables.

10. The method of claim 9 further comprising defining a minimum resolution of the programming stage defined by the difference between the critical section of the shared variables and the arrival time of a subsequent packet wherein the critical section is less than the arrival time of the subsequent packet.

11. The method of claim 4 wherein each of the plurality of programming engines executes a plurality of contexts simultaneously.

12. The method of claim 11 wherein the plurality of contexts are executed in an order.

13. The method of claim 12 wherein the order includes a read phase and a write-modify phase.

14. The method of claim 13 wherein the CAM includes a plurality of entries for monitoring least recently used variables.

15. The method of claim 14 wherein the read phase includes determining the cache status of a shared variable in the CAM and updating a value for the shared variable if the shared variable

is cached in the CAM.

16. The method of claim 14 wherein the read phase executes a read of the shared variable from a local memory in at least one of the plurality of contexts.

17. The method of claim 16 wherein the remaining plurality of contexts hide a latency time of the read by executing other assigned tasks for processing packets.

18. The method of claim 16 wherein the shared variable is cached in the CAM and is available for processing at the modify-write phase in the at least one of the plurality of contexts.

19. A processor comprising:

a plurality of programming engines including:

specific tasks for packet processing assigned to the plurality of programming engines;

programming stages corresponding to the plurality of programming engines;

a plurality of pipelines between the programming stages; and

a plurality of contexts corresponding to the plurality

of programming engines for the assigned tasks.

20. The processor of claim 19 wherein each of the plurality of programming engines further includes next neighbor registers for transferring data from a next neighbor register residing in a currently executing programming engine to a subsequent next neighbor register residing in an adjacent programming engine.

21. The processor of claim 20 wherein each of the plurality of programming engines further includes a content addressable memory (CAM).

22. The processor of claim 21 wherein the CAM includes a plurality of entries for monitoring least recently used variables.

23. The processor of claim 21 further comprising shared variables utilized by the programming stages of the plurality of programming engines, the shared variables including a critical section defining the read-modify-write time of the shared variables.

24. The processor of claim 23 further comprising a minimum

resolution of the programming stage defined by the difference between the critical section of the shared variables and the arrival time of a subsequent packet wherein the critical section is less than the arrival time of the subsequent packet.

25. The processor of claim 21 wherein the plurality of contexts are executed in order, the order including a read phase and a write-modify phase, wherein the read phase includes determining the cache status of a shared variable in the CAM and updating a value for the shared variable if the shared variable is cached in the CAM, and wherein the read phase executes a read of the shared variable from a local memory in at least one of the plurality of contexts.